DDoS attack detection and defense in SDN based on machine learning

Tan-Khang Luong*, Trung-Dung Tran*, Giang-Thanh Le*
* Faculty of Information Technology, University of Science, Ho Chi Minh city, Vietnam.
Vietnam National University, Ho Chi Minh city, Vietnam.

2020 7th NAFOSTED Conference on Information and Computer Science (NICS) | 978-0-7381-0553-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/NICS51282.2020.9335867

***Abstract*—Distributed Denial of Service (DDoS) attack is one of the most dangerous threats in computer networks. Hence, DDoS attack detection is one of the key defense mechanisms. In this paper, we propose a DDoS detection and defense approach in Software Defined Network (SDN) systems based on machine learning (ML) and deep neural network (DNN) models. The combination of ML and DNN classifiers with the centralized factors of SDN can efficiently mitigate the harmful effect of DDoS to the network system. Besides, we conducted two types of attack scenarios, one is from inside and one is from outside of the network system.**

***Index Terms*—DDoS, Machine Learning, SDN**

I. Introduction

DDoS attack is the fatal and widespread threat on today's Internet. Massive DDoS attacks appear more frequently. In 2016, the world witnessed one of the largest DDoS attacks at that time. Dyn, a DNS service provider company, experienced a massive DDoS attack, at its peak, the system suffered incoming traffic at a rate of 1.2Tbps [1]. This particular attack was based on a botnet (Mirai) network on unsecured IoT devices which were allowed non-authorized attackers accessing remotely. In 2018, Github experienced the largest DDoS attack in history, at a rate of 1.35Tbps. The attackers launched an amplification based on exploiting the unsecured Memcached servers [2]. Moreover, with the combination of multiple attack vectors, the conventional statistical-based methods show their weakness to identify abnormal traffic. An attack with a small volume can even be seen as a normal one in the early stages, therefore, statistical-based methods were not efficient to low-rate attacks. Besides, machine learning methods identify DDoS based on statistical features performed better than statistical traffic approaches.

SDN is the promising network architecture that allows logical programming with an abstraction level behind the network's operation. SDN commonly includes three parts, the OpenFlow switch, the host, and the controller. The most important part of SDN is the controller. It is responsible for the generation, delivery, maintenance of the network forwarding flow table. The OpenFlow switch is another core component and mainly responsible for routing and forwarding data packets [3].

Our work proposes a DDoS attack classifier based on machine learning and deep neural network to interact with our conducted SDN controller to intermediately drop the harmful flows from both inbound and outbound traffic. Moreover, we also indicate the gap between theoretical results and real experiment results.

The rest of this paper is structured as: Section II discusses several related works, section III describes our proposed approach and experiment results, and the conclusion is in section IV.

II. Related works

A. Statistical approaches to DDoS detection

Statistical approaches to DDoS detection are the conventional methods, which are based on monitoring the entropy variations of header fields in packets. In the early 2000s, this approach was proposed with the expectation that during a volumetric attack, the randomness of traffic features is subject to sudden variations [4]. The bandwidth attacks are defined by a huge set of compromised devices that send a high volume of traffic to one or multiple victims. As a result, these traffic caused a rise (or a drop) in the distribution of some attributes in packets' header, eg. the decreasing of target IP and the increasing of source IP. Therefore, DDoS is usually identified based on the average of threshold values on these distribution indicators.

Feinstein *et al.* [5] proposed the earliest method to detect DDoS based on Chi-square distribution and computation of source IP address entropy. The authors indicated that the variation of source IP and Chi-square statistics in abnormal traffic was massively greater than that one in legitimate traffic. Similarity to this, P.Bojovic *et al.* [6] observed the bandwidth of traffic entropy variation to detect high-rate DDoS attacks.

A usual limitation of these approaches is that entropy-based techniques need to adopt the exact threshold value. In different network systems, the deviation of traffic volume is completely various leads to a challenge to find out adaptive methods, which could generate a high appropriate threshold to minimize both false positive and negative detection. Kumar *et al.* [7], proposed an adaptive technique to actively adapt the threshold in different network systems.

B. Machine and deep learning approaches to DDoS detection

Several datasets related to DDoS fields have been released recently. Before 2012, most of the datasets were small and not diverse in attack methods. Some of them are DARPA [8], KDD CUP 99 [9], NSL-KDD [10], or a private dataset - Booster [11]. However, after 2012, the Canadian Institute for Cybersecurity in University of New Brunswick released three large-scale datasets in computer security in 2012, 2017, and 2018 called ISCXIDS2012 [12], CIC-IDS2017 [13], and CSE-CIC-IDS2018 on AWS [14] respectively. CSE-CIC-IDS2018 on AWS (CICIDS2018) is the most novel and noblest network security dataset at this time. The dataset was constructed on a simulation network conceived on AWS clouds includes hundreds of computers, servers, and network devices.

Most machine learning-based approaches used popular algorithms such as Support Vector Machine (SVM), Linear Regression (LR), Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), etc. Among these methods, SVM with linear kernel always showed high accuracy and stability on different datasets and network systems. He *et al.* [15], used nine machine learning algorithms involves LR, SVM (linear, RBF, Polynomial kernel), DT, NB, RF, K-means, Gaussian EM. In the experiment, Linear SVM produced the best accuracy (99.7%) with the lowest false cases (less than 0.07%). Similarly, R.Doshi *et al.* [16] applied many ML algorithms to detect DDoS from the origin of attacks in IoT systems. The authors mainly observed the high volumetric traffic when the IoT network was intruded. As a result, the Linear SVM model also reported achieving a high accuracy at 99.1%.

Some research using deep learning showed the potential of earlier detection of DDoS. DL approaches commonly rely on two basic model architectures are CNN and RNN. Some of them rely on the abilities of learning from time-series data of RNN and others based on the sliding-window technique in CNN to extract the features of traffic. In the research [17], [18], [19] the authors used RNN models. Yuan *et al.* [17], used both RNN (LSTM, GRU) and the combination of CNN and RNN and trained on the dataset ISCXIDS2012. They reported an extremely high result compared to the common ML algorithms Random forest. Meanwhile, [4], [20] used CNN models. R. Doriguzzi-Corin *et al.* [4] realized a limitation of RNN architecture is high computation cost. Therefore, they used Convolution1D and MaxPooling layers in CNN to simplify the architecture but still keep the accuracy. Basnet *et al.* [21] tried to measure the performance of different deep learning frameworks on a deep neural network to detect DDoS. The authors train the model on the dataset CICIDS2018 with many DL frameworks such as Theano, Tensorflow, FastML, etc.

C. DDoS defense in SDN

In SDN, several methods introduced to detect and mitigate the effect of DDoS attacks. Kim *et al.* [22] proposed a technique to predict harmful bandwidth based on the threshold of flow. The author applied Cisco's NetFlow Technology to detect network traffic by the extracted flow features and pre-defined thresholds. Manso *et al.* [23] applied an open-source intrusion detection system (IDS) called SNORT to early notify attacks to SDN controllers by setting a combination of rules. The main idea of the authors is to detect the DDoS from the source of attacks. Niyar *et al.* [24] proposed a deep neural network model based on Stacked Autoencoder to detect and mitigate DDoS in SDN. Firstly, the authors conducted a simulation of SDN to collect samples of legitimate and malicious traffic to build a dataset. Their dataset included the most common attacks such as SYN flood, UDP flood, TCP flood, etc. The goal of the authors is to detect multi-vector attacks. In the experiment, the result was reported higher than 90% in most scenarios and up to 99% in binary classification. Li *et al.* [3] applied the RNN models in [17] to extract traffic features. They conducted a multi-module defense system. The traffic features are stored and weighted in a statistical module, and when these features meet a pre-set threshold, a notification module will fire a command to the SDN controller to drop the flow between two IPs.

III. Proposal approach

In this section, first of all, we conduct the machine learning classifier with Linear Support Vector Machine (LSVM), Decision Tree, Random Forest, and Naive Bayes algorithms on the data set CSE-CIC-IDS2018. Besides, we also build a deep neural network-based classifier to compare with the above conventional ML methods. Secondly, we conduct a DDoS detection software called IDS-DDoS that collects the network packets to generate and classify flows and notify the malicious ones to the SDN controller. Thirdly, we build an SDN controller that receives abnormal flow identification to produce dropping rules in forwarding flow tables on OpenFlow switches. Finally, we experiment with two attack scenarios on a simulation network.

A. Machine learning and deep neural network classifier

1) Dataset: We train our models on the dataset CICIDS2018. This dataset was produced with an open-source software called CICFlowMeter [25] on a series of large size PCAP files. CICFlowMeter generates 84 statistical attributes of traffic flow, for example, source IP, destination IP, flow duration, max, min, mean, standard deviation values of packets' size, etc. We choose the parts in the dataset containing DDoS traffic only. The DDoS traffic was conducted by well-known DDoS attack tools such as Hulk, Golden-Eyes, HOIC, etc.

There are some error data points containing NaN values, negative values. Thus we have to remove these records from the dataset. Finally, the dataset remains more than 11 million data points. After that, we split the dataset into two sets, one for training and another for testing called IDS-Train and IDS-Test respectively.

Finally, we apply feature selection with the Chi-square test to find the best features for training on IDS-Train. After all, we keep 67 features in total. The top 5 features are "No. packets have data in forward flow", "Total length of forward packets", "Total backward packets per second", "Window bytes of initial forward flow", "Average segmentation size in forward flow".

Before training, we normalize the IDS-Train using L2-norm. We do not scale the training set, which could lead to overfitting.

2) Machine Learning and Deep Neural Network models: We apply four machine learning models including Linear SVM, Naive Bayes, Decision Tree, and Random Forest. We use Python programming language with scikit-learn library and keep all default parameters.

To find the best deep neural network model. First of all, we build a series of models from 2 to 6 hidden fully connected layers, with 32 units each layer. After that, we increase the number of units in each layer of the 6-layer model from 32 to 64 and 128 units. After achieving the highest accuracy, we start reducing the number of hidden layers from 6 to 2. Finally, we got the simplest model with the best accuracy shows in figure 1. To train our deep neural network models, we use Keras library with Tensorflow backend and keep all default parameters for dense layers. The training pipeline includes 10 epochs with a batch size of 2000 data points and we keep the best accuracy models on validation set only.

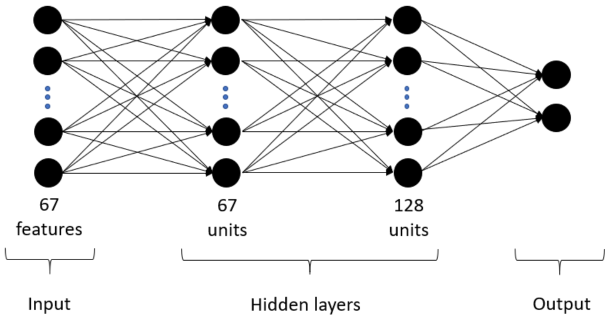


Fig. 1. DNN model

B. Result on IDS-Test

We assess the models with 4 conventional assessment indicators including accuracy, precision, recall, and f1-score. We define binary-classification with malicious flow as Positive and legitimate flow as Negative. The formula of the above indicators is as below, with TP, FP, TN, and FN denotes True Positive, False Positive, True Negative, and False Positive respectively.

Accuracy:

$$Acc = \frac{TP + TN}{Total}$$

Precision (or Positive predictive value):

$$Pre = \frac{TP}{TP + FP}$$

Recall:

$$Rec = \frac{TP}{TP + FN}$$

F1-score:

$$F1 = 2\frac{Pre \cdot Rec}{Pre + Rec}$$

The testing result is shown in the table I. Decision Tree is an algorithm with a naive and simple approach, however it performs the best result among these models, even higher than the most complex model DNN. It shows that the more complex architecture might not ensure the more accurate.

TABLE I
RESULT ON IDS-TEST

Model	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
LSVM	95.67	88.05	86.91	87.48
NB	67.69	34.92	99.31	51.67
DT	**99.97**	**99.91**	**99.94**	**99.92**
RF	99.83	99.11	99.94	99.52
DNN	99.22	97.91	97.58	97.74

C. IDS-DDoS

In this section, we apply our models in the previous section to conduct a software that has the ability to capture, collect packets to generate traffic flow, and classify the traffic flow as malicious or legitimate, called IDS-DDoS.

We conduct the software with Python programming language. On the part of the capturing and collecting packets, we use the library called Scapy. To generate traffic flow, we refer to the source code of CICFlowMeter [1] which was written in Java. The workflow of IDS-DDoS is explained in figure 2. The socket server in this scenario is an SDN controller.

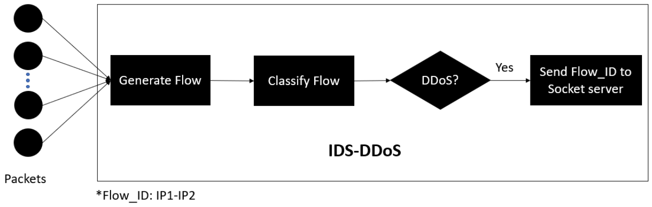


Fig. 2. IDS-DDoS workflow

D. SDN controller

We apply Ryu framework to conduct an SDN controller. Our controller holds two different responsibilities, one is as a normal SDN to control and monitor the network and one is as a defend layer. The workflow of this SDN is presented in figure 3. We apply the simplest threshold measurement method which indicates a traffic flow as malicious flow when its identification is sent to the controller N times within T seconds.

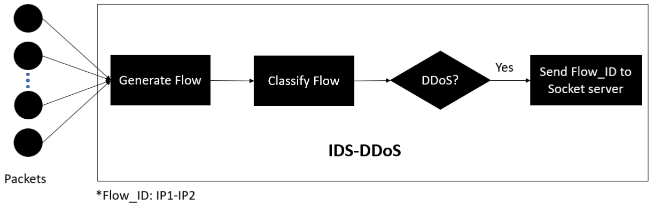
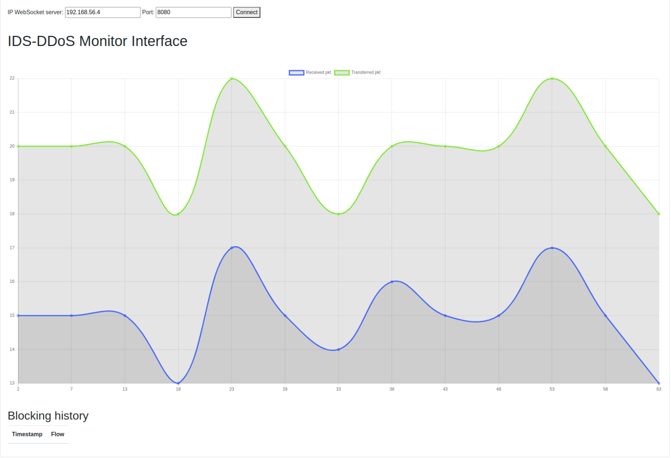


Fig. 3. SDN controller workflow

To show the traffic monitor in SDN, we conduct a visual graphic interface as in figure 4. Y-axis indicates the number of packets, X-axis indicates timestamp, the blue line indicates the

[1] https://github.com/ahlashkari/CICFlowMeter

number of received packets, and the green line indicates the number of transferred packets. Below the graph is “Blocking history” containing a list of blocked Flow_ID.


IP WebSocket server: 192.168.56.4 Port: 8080 Connect

IDS-DDoS Monitor Interface

Blocking history

Timestamp	Flow

Fig. 4. SDN monitor GUI

E. Experiment on simulation network

1) Simulation network: The simulation network is shown in figure 5.

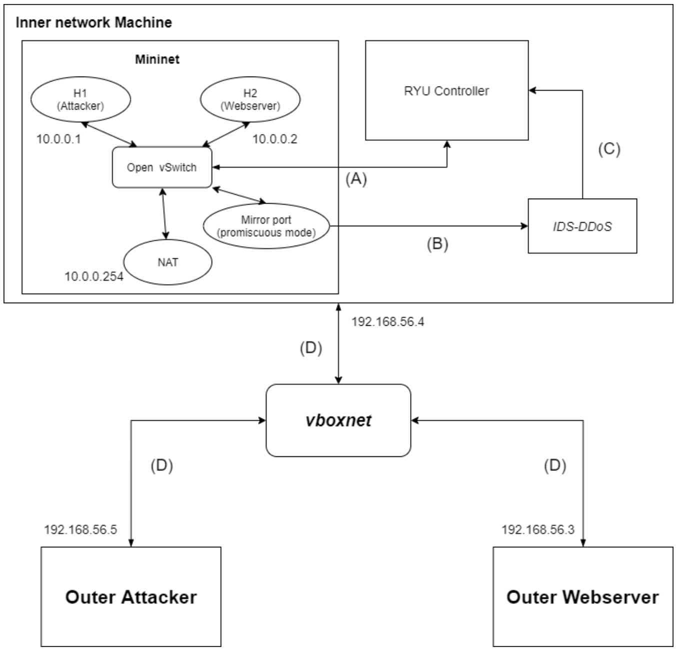


Fig. 5. Simulation network

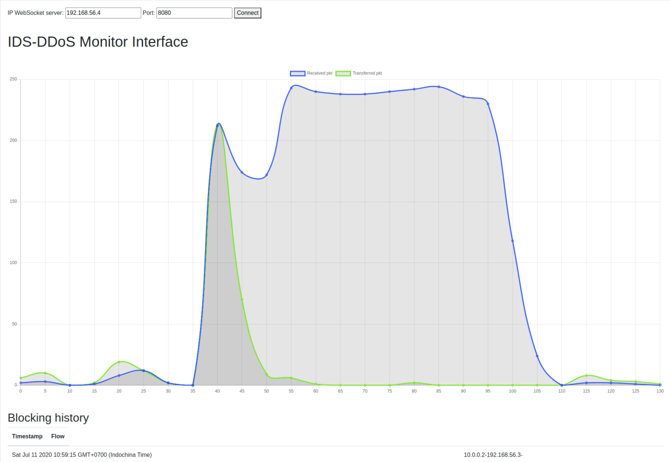
There are 3 virtual machines “Inner network machine”, “Outer attacker”, and “Outer Webserver”. These machines are connected to each others via a “Host Only” of Virtualbox called “vboxnet”. The vboxnet represents the Internet, the outer attacker and the outer webserver represents random attackers and webservers on the Internet respectively, and the inner network machine represents our local network. Inner network machine contains 3 parts, the first part is the SDN controller, the second part is IDS-DDoS software, and the third one is a simulation of SDN created with Mininet. The simulation SDN contains 2 machines called H1 and H2. These machines communicate with the outside world through a NAT device. H1, H2 represents the inner attacker and inner webserver respectively.

2) Experiment results: To launch the experiment, we propose two scripts of attack scenarios. In the first scenario, our SDN system is attacked by a random attacker on the Internet. In the second scenario, on the other hand, our local network is intruded on, and the devices in SDN become a part of a botnet attacking a random web server on the Internet.

For the first script, we launch the HOIC attack tool on the outer attacker to attack H2. For the second one, we apply the Hulk attack tool on H1 to attack the outer webserver. In these scripts, we set the threshold as “appearing 10 times within 1 minute”.

The model we use to classify the traffic flow is Linear SVM. Theoretically, Decision Tree and Random Forest achieve extremely high accuracy. Nonetheless, in the experiment, they could barely recognize the malicious flows. Meanwhile, DNN and Linear SVM recognize the abnormal flows quite well. Besides, Linear SVM run faster than DNN more than 80 times on CPU. Therefore, we decide to use Linear SVM instead of DNN.

a) Script 1: Figure 6 shows the state of the SDN before, during, and after the attack. Before the attack begins, the state of the network is stable, but when the attack is launched, the number of received and transferred packets suddenly increases. However, after a while, when the IDS-DDoS classifies enough flow as malicious ones, and the SDN controller matched the Flow_ID with a defined threshold and dropped the flow, the number of transferred packets drops dramatically.


IP WebSocket server: 192.168.56.4 Port: 8080 Connect

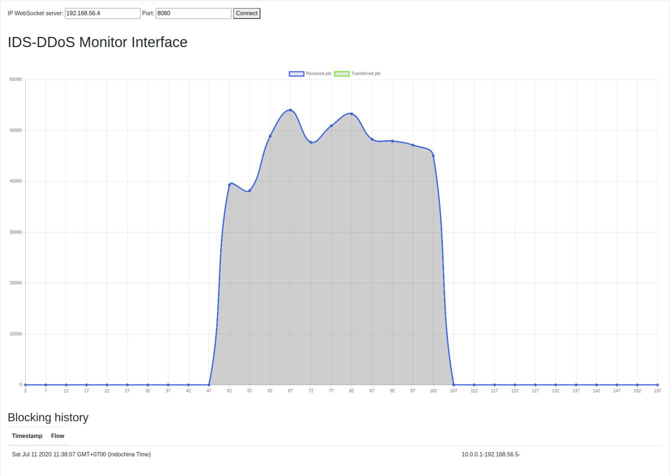
IDS-DDoS Monitor Interface

Blocking history

Timestamp	Flow
Sat Jul 11 2020 10:59:15 GMT+0700 (Indochina Time)	10.0.0.2-192.168.56.3-

Fig. 6. Script 1

b) Script 2: Figure 7 presents the state of the SDN before, during, and after the attack. Similarly to the previous script, after about 1 minute, the system recognizes and drops the flow. However, on script 1, after the flow is dropped, the number of received packets still increases while the number of transferred packets decreases dramatically. Meanwhile, on script 2, the number of both received and transferred networks drops dramatically. The reason is, on script 1, when the SDN prevents the communication between the outer attacker and H2, the attacker still keeps sending new packets going through the monitor, on the other hand, on script 2, all packets are sent from H1 is dropped before it travels through the monitor.



IP WebSocket server: 192.168.56.4 Port: 8080 Connect

IDS-DDoS Monitor Interface

Blocking history

Timestamp	Flow
Sat Jul 11 2020 11:38:07 GMT+0700 (Indochina Time)	10.0.0.1-192.168.56.5-

Fig. 7. Script 2

IV. Conclusion

- Decision Tree and Random Forest are the best model on simulation, but in practice, the Linear SVM and DNN are outperforming. There is a need for further research to fill the gap between theory and practice. Through the experiments, we observe that the more complex DDoS detection system might not produce more accurate results than the simple one.
- Our proposed models are sensitive to the new attacks, because it is trained with the dataset collected from 8 different attacks only. However, with the simple architecture of SVM, the system can easily learn and classify any new attacks.

References

[1] "What we know about friday's massive east coast internet outage." https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/, [Accessed 02-Apr-2020].

[2] "Github survived the biggest ddos attack ever recorded." https://www.wired.com/story/github-ddos-memcached/, [Accessed 02-Apr-2020].

[3] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of ddos attack–based on deep learning in openflow-based sdn," *International Journal of Communication Systems*, vol. 31, no. 5, p. e3497, 2018. e3497 IJCS-17-0848.R1.

[4] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[5] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 303–314 vol.1, 2003.

[6] P. Bojović, I. Bašičević, S. Ocovaj, and M. Popović, "A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method," *Computers & Electrical Engineering*, vol. 73, pp. 84–96, 2019.

[7] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[8] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, p. 579–595, Oct. 2000.

[9] S. Stolfo, "Kdd cup 99 dataset." http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, [Accessed 02-Apr-2020].

[10] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[11] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters — an analysis of ddos-as-a-service attacks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 243–251, 2015.

[12] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, p. 357–374, May 2012.

[13] I. Sharafaldin., A. H. Lashkari., and A. A. Ghorbani., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pp. 108–116, INSTICC, SciTePress, 2018.

[14] C. I. for Cybersecurity, "A realistic cyber defense dataset (cse-cic-ids2018)." https://registry.opendata.aws/cse-cic-ids2018/, [Accessed 02-Apr-2020].

[15] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 114–120, 2017.

[16] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, 2018.

[17] X. Yuan, C. Li, and X. Li, "Deepdefense: Identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8, 2017.

[18] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, "A new multi classifier system using entropy-based features in ddos attack detection," in *2018 International Conference on Information Networking (ICOIN)*, pp. 162–167, 2018.

[19] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[20] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, pp. 1–9, 07 2018.

[21] C. J. L. W. Ram B. Basnet, Riad Shash and T. Doleck, "Towards detecting and classifying network intrusion traffic using deep learning frameworks," *Journal of Internet Services and Information Security (JISIS)*, vol. 9, pp. 1–17, nov 2019.

[22] Myung-Sup Kim, Hun-Jeong Kong, Seong-Cheol Hong, Seung-Hwa Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507)*, vol. 1, pp. 599–612 Vol.1, 2004.

[23] J. S. C. Manso, Pedro; Moura, "Sdn-based intrusion detection system for early detection and mitigation of ddos attacks," *MDPI Information journal*, vol. 10, p. 106, mar 2019.

[24] Q. Niyaz, W. Sun, and A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," *arXiv e-prints*, p. arXiv:1611.07400, Nov. 2016.

[25] C. I. for Cybersecurity, "Cicflowmeter." https://www.unb.ca/cic/research/applications.html, [Accessed 02-Apr-2020].